Kotlin - Abstract Classes

A Kotlin abstract class is similar to Java abstract class which can not be instantiated. This means we cannot create objects of an abstract class. However, we can inherit subclasses from a Kotlin abstract class.

A Kotlin abstract class is declared using the **abstract** keyword in front of class name. The properties and methods of an abstract class are **non-abstract** unless we explictly use **abstract** keyword to make them abstract. If we want to override these members in the child class then we just need to use **override** keyword infront of them in the child class.

```
abstract class Person {
   var age: Int = 40

   abstract fun setAge()    // Abstract Method

   fun getAge() {           // Non-Abstract Method
      return age
   }
}
```

> Abstract classes are always open. You do not need to explicitly use **open** keyword to inherit subclasses from them.

Example

Following is a simple example showing a Kotlin Abstract class and its implementation through a child class:

```
abstract class Person(_name: String) {
   var name: String
   abstract var age: Int

   // Initializer Block
   init {
      this.name = _name
   }

   abstract fun setPersonAge(_age:Int)
   abstract fun getPersonAge():Int

   fun getPersonName(){

       println("Name = $name")
```

mohamedsohel.co.in

```
    }
}
class Employee(_name: String): Person(_name) {
    override var age: Int = 0

    override fun setPersonAge(_age: Int) {
        age = _age
    }
    override fun getPersonAge():Int {
        return age
    }
}
fun main(args: Array<String>) {
    val employee = Employee("Zara")
    var age : Int

    employee.setPersonAge(20)

    age = employee.getPersonAge()

    employee.getPersonName()
    println("Age = $age")
}
```

When you run the above Kotlin program, it will generate the following output:

```
Name = Zara
Age = 20
```

Here, a class **Employee** has been derived from an abstract class **Person**. We have implemented one abstract property and two abstract methods in the child class **Employee**. Here notable point is that all the abstract members have been overriden in the child class with the help of **override**, which is a mandatory for a child class if it inherits an abstract class.

To summarise, A Kotlin class which contains the **abstract** keyword in its declaration is known as abstract class.

- Abstract classes may or may not contain *abstract methods*, i.e., methods without body (public void get();)
- But, if a class has at least one abstract method, then the class **must** be declared abstract.
- If a class is declared abstract, it cannot be instantiated.
- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

mohamedsohel.co.in

Quiz Time (Interview & Exams Preparation)

Q 1 - Which one is true about a Kotlin abstract class :

A - An abstract class can have abstract and non-abstract members

B - An abstract class can not be instantiated.

C - An abstract class can be inherited by a child class

D - All the above

Q 2 - Which keyword is used to implement the abstract members of a abstract class:

A - init

B - override

C - Override

D - None of the bove

AD

mohamedsohel.co.in